333-CD-003-005

EOSDIS Core System Project

Release A SCF Toolkit Users Guide for the ECS Project

November 1996

Hughes Information Technology Systems
Upper Marlboro, Maryland

Release A SCF Toolkit Users Guide for the ECS Project

November 1996

Prepared Under Contract NAS5–60000
CDRL Item 062

SUBMITTED BY

R.E. Clinard /s 11/22/96

Robert E. Clinard, ECS CCB Chairman Date
EOSDIS Core System Project

Hughes Information Technology Systems
Upper Marlboro, Maryland

333-CD-003-005

This page intentionally left blank.

Preface

This SCF Toolkit version is directed at EOS instrument data providers who will deliver code to the ECS Release A DAACs. It is an engineering upgrade to Toolkit 5, delivered in August, 1995. The user calling interface of the current version is the same as that of Toolkit 5, except that several new functions have been added and internal interfaces have been made compatible with Release A.

Note: Users who implement the current Toolkit version, are not guaranteed to be able to test their PGEs at the Ir1 DAACs. These are at EDC, GSFC and LaRC. Release A production system capability will not be available until Release A Science Software Integration and Test activities begin, in the first quarter of 1997.

This document is a formal contract deliverable with an approval code 1. It requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once this document is approved, Contractor approved changes are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

The purpose of this document is to provide Earth Observing System (EOS) instrument data processing software developers and scientists, knowledge of Toolkit functionality, provide a listing of routine calling sequences, detailed descriptions and examples of usage. The Toolkit will be used by developers at their Science Computing Facilities (SCFs) to develop EOS data production software and to prepare that software for integration into distributed active archive centers (DAACs). Subsequent usage of the Toolkit will be in conjunction with the services provided by the DAACs to produce, archive and distribute standard products. This document accompanies a software delivery that contains implementations of the tools described in the document. We note that this version of the Toolkit contains provisions for error/status message, process control and file name handling by science software in lieu of an operational scheduling. This handling will be via manual manipulation of UNIX files. This version also contains tools for creation and access of standard product metadata parameters as well as several added ancillary data files (e.g., a geoid model).

The hierarchical data format (HDF) has been selected by the Earth Observing System Data and Information System (EOSDIS) Project as the format of choice for standard product distribution. HDF is a *disk format* and *subroutine library* for storage of most kinds of scientific data. As a *disk format*, HDF files consist of a directory and an unordered set of binary data objects. Each directory entry describes the location, the type, and the size of these binary objects.

The *HDF subroutine library* is designed to be easy for C and FORTRAN programmers to use. The HDF library consists of callable routines, each of which belongs to a particular *interface*. Each interface within these layers address a particular HDF function or a particular HDF data structure, such as arrays, tables, and annotations.

This Users Guide is accompanied by a Toolkit Primer. The Primer is intended to provide a concise explanation of individual tool usage, functionality and coding examples. The Primer will not contain details, appendices, requirements trace, and so on; that are contained in this Users Guide. The Primer is available on the World Wide Web (WWW). The universal reference locator (URL) for the EOSDIS Core System (ECS) Data Handling System (EDHS) home page is:

http://edhs1.gsfc.nasa.gov/

If you have a WWW browser such as Mosaic or Lynx, you can view this document. Under

"ECS Development"

"Toolkits"

click on

"SDP Toolkit Primer".

A list of Frequently Asked Questions (FAQ) for Toolkits is also available.

The URL for the SDP Toolkit Frequently Asked Questions (FAQ) page is http://newsroom.hitc.com/sdptoolkit/faq.html

You can also get there from the EDHS Home Page http://edhs1.gsfc.nasa.gov. Click on the "Toolkit Frequently Asked Questions (FAQ)" link from the SDP Toolkit Page.

The technical point of contact within the EOS Data and Information System (EOSDIS) core System (ECS) project is:

Larry Klein, ARC Toolkit Manager: larry@eos.hitc.com

An e–mail drop for user questions and comments is: pgstlkit@eos.hitc.com

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Systems
1616 McCormick Drive
Upper Marlboro, Maryland 20774-5372

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Submitted as Final
iii through xxiv	Submitted as Final
1–1 through 1–12	Submitted as Final
2–1 and 2–2	Submitted as Final
3–1 and 3–2	Submitted as Final
4–1 through 4–8	Submitted as Final
5–1 through 5–22	Submitted as Final
6–1 through 6–476	Submitted as Final
A–1 through A–10	Submitted as Final
B–1 through B–4	Submitted as Final
C–1 through C–60	Submitted as Final
D–1 through D–18	Submitted as Final
E–1 through E–16	Submitted as Final
F–1 through F–10	Submitted as Final
G–1 through G–14	Submitted as Final
H–1 and H–2	Submitted as Final
I–1 through I–14	Submitted as Final
J–1 through J–58	Submitted as Final
K–1 through K–6	Submitted as Final
L–1 and L–8	Submitted as Final
M-1 and M-2	Submitted as Final
AB–1 through AB–6	Submitted as Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
333-CD-002-001	Original	February 1995	95-0148
333-CD-002-002	Original	March 1995	95-0195
333-CD-003-001	Original	July 1995	95-0491
333-CD-003-002	Original	August 1995	95-0595
333-CD-003-003	Original	May 1996	96-0419
333-CD-003-004	Submitted as Final	May 1996	96-0513A
333-CD-003-005	Submitted as Final	November 1996	96-1314

This page intentionally left blank.

Contents

1. Introduction

2. Related Documentation

3. Toolkit Design Goals

4. Toolkit Usage, Functionality and Future Direction

5. Toolkit Installation and Maintenance

6. SDP Toolkit Specification

Appendix A. Assumptions

Appendix B. Status Message File (SMF) Creation and Usage Guidelines

Appendix C. Process Control Files

Appendix D. Ancillary Data Access Tools

Appendix E. Example of Level 0 Access Tool Usage

Appendix F. Level 0 File Formats

Appendix G. PGS_GCT Information Relating To Interface Specification

Appendix H. PGS_CUC_Cons - Example Standard Constants File

Appendix I. PGS_CUC_Conv—Input File Provided With the UdUnits Software

Appendix J. Population of Granule Level Metadata Using the SDP metadata tools

Appendix K. POSIX Systems Calls Usage Policy

Appendix L. Ephemeris And Attitude File Formats

Appendix M. Problem Identification List

Abbreviations and Acronyms

List of Figures

List of Tables

This page intentionally left blank.

1. Introduction

1.1 Identification

The *SDP Toolkit Users Guide for the ECS Project* is a part of the Science Data Production (SDP) Toolkit delivery made under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract (NAS5–60000). It was first delivered in January 1994. The current Users Guide is updated for the Release A Toolkit delivery made in November 1996. The *SDP Toolkit Users Guide for the ECS Project* will be updated for each major release of the SDP Toolkit.

1.2 Scope

This SCF Toolkit version is directed at EOS instrument data providers who will deliver code to the ECS Release A DAACs. It is an engineering upgrade to Toolkit 5, delivered in August, 1995. The user calling interface of the current version is the same as that of Toolkit 5, except that several new functions have been added and internal interfaces have been made compatible with Release A.

Note: Users who implement the current Toolkit version, are not guaranteed to be able to test their PGEs at the Ir1 DAACs. These are at EDC, GSFC and LaRC. Release A production system capability will not be available until Release A Science Software Integration and Test activities begin, in the first quarter of 1997.

The SDP Toolkit Users Guide describes Toolkit routine usage for science software developers, who will produce code to process instrument data. These routines have been designed to satisfy the requirements in the SDP Toolkit Requirements Specification for the ECS Project (October 1993) and revisions (July 1995). The current version of the Users Guide is for the Release A Toolkit delivered code, however, the Toolkit will be updated as requirements are updated, certified and requirements for later platform instruments are determined. This document describes the overall design of the Toolkit, provides a general explanation of usage, and installation procedures on computer platforms for which software development and certification have been done. Detailed listings of routines, calling sequences, inputs and outputs and examples of usage are also provided.

1.3 Purpose and Objectives

This document is aimed at the EOS data production software developers and scientists who will use the SDP Toolkit to encapsulate their code in the distributed active archive center (DAAC) computing facilities. The purpose of the Toolkit is to provide an interface between instrument processing software and the production system environment. It sets up the context and environment to facilitate portability of code for the execution of production processes and the transfer of data sets and information to those processes. This interface will be implemented in the Science Computing Facility (SCF) development environment, along with additional utilities that will be used to emulate production environment services.

An important goal of the Toolkit is to facilitate the smooth transition and integration of code into the DAAC by abstracting out science process dependencies on external system architecture. Another goal is the provision of an interface into which application modules can be incorporated. This may include, for example, math packages; other specialized routines that can be commercial–off–the–shelf software (COTS); freeware; or user supplied modules. An effort will be made during development to incorporate and reuse existing application software modules.

This Users Guide will lay out the high level design of the Toolkit and provide sufficient description of routines to show how EOS science software should incorporate the Toolkit interface.

In the description of the Toolkit routines, descriptive information is presented in the following format:

TOOL TITLE

NAME:	Procedure or routine name
SYNOPSIS:	
C:	C language call
FORTRAN:	FORTRAN77 or FORTRAN90 language call
DESCRIPTION:	Cursory description of routine usage
INPUTS:	List and description of data files and parameters input to the routine
OUTPUTS:	List and description of data files and parameters output from the routine
RETURNS:	List of returned parameters indicating success, failure, etc.
EXAMPLES:	Example usage of routine
NOTES:	Detailed information about usage and assumptions
REQUIREMENTS:	Requirements from *PGS Toolkit Specification*, Oct. 93 which the routine satisfies

1.4 Status and Schedule

The SDP Toolkit Users Guide is based on requirements contained within the *PGS Toolkit Requirements Specification* delivered to the ECS Project in October 1993 and updated in July 1995. This Users Guide accompanies a set of toolkit routines, delivered in April 1996. Table 1–2 below gives a complete listing; brief description; and delivery dates of Toolkit software available to users. We note also several important related schedule items:

- April 1995—IDL was selected as the Toolkit graphics package of choice. A memo describing distribution to IDL software and documentation access will be sent to instrument teams at about the time of delivery of this toolkit.
- July 1995—Release Toolkit A delivery, including prototype HDF-EOS swath structure software
- July 1995— Delivery (to the EOS community) of a draft HDF-EOS standard and users guide.

- ≠ January 1996—ECS Interim Release 1 (Ir1). This release to Goddard Space Flight Center (GSFC), Earth Resources Observation Systems (EROS) Data Center (EDC) and Langley Research Center (LaRC) DAACs, includes the Toolkit, and other software for Tropical Rainfall Measuring Mission (TRMM) algorithm integration and test and instrument support processing. We will call this the DAAC version of the Toolkit.
- May 1996— Release A SCF Toolkit delivery.
- July 1996 HDF-EOS version 1.0 delivery
- November 1996 updated HDF-EOS and SCF Toolkit delivery

Table 1–1 provides a key to the tool names and the section where the specific tools can be located.

Table 1–1. Toolkit Routine Key

Key	Class	Section
AA	Ancillary Data Access	6.3.1
CBP	Celestial Body Position	6.3.2
CSC	Coordinate System Conversion	6.3.3
CUC	Constant and Unit Conversions	6.3.6
EPH	Ephemeris Data Access	6.2.6
GCT	Geo Coordinate Transformation	6.3.4
IO	Input Output (File I/O)	6.2.1
MEM	Memory Management	6.2.4
MET	Meta Data Access	6.2.1
PC	Process Control	6.2.3
SMF	Status Message File (Error/Status)	6.2.2
TD	Time Date Conversion	6.2.7

In Table 1–2 a list of Toolkit routines is given, with delivery data and page number references in this Users Guide.

Table 1–2 lists Toolkit routines alphabetically by class as defined in the key below. The class keyword follows the Product Generation System (PGS) keyword (i.e., PGS_AA).

Table 1–2. Toolkit Routine Listing (1 of 7)

Tool Name	Description	Date	Page
pccheck	Use to verify that a process control file (PCF) is syntactically correct	10–94 7–95	6–191
PGS_AA_2Dgeo	Allows access to 2 dimensional data sets, e.g., sea–ice	10--94 2–95 7–95 4-96	6–295
PGS_AA_2DRead	Allows access to 2 dimensional data sets, e.g., sea–ice	10--94 2–95 4-96	6–305
PGS_AA_3Dgeo	Allows access to 3 dimensional data sets,e.g., atmospheric humidity	10--94 2–95 4-96	6–300
PGS_AA_3DRead	Allows access to 3 dimensional data sets,e.g., atmospheric model	10--94 2–95 4-96	6–310

Table 1–2. Toolkit Routine Listing (2 of 7)

Tool Name	Description	Date	Page
PGS_AA_dcw	Returns the surface types (land, sea, coast), and nation–state to be determined (TBD) for a user defined set of locations	10--94 4-96	6–281
PGS_AA_dem	Locates heights from specified digital elevation model (DEM) corresponding to each of the locations specified	2–95 7–95 4-96	6–285
PGS_AA_PeVA_integer	Searches in a specified file for the parameter and returns the value of that parameter which is an integer	10–94 2–95 7–95 4-96	6–293
PGS_AA_PeVA_real	Searches in a specified file for the parameter and returns the value of that parameter which is a real(float)	10–94 2–95 7–95 4-96	6–291
PGS_AA_PeVA_string	Searches in a specified file for the parameter and returns the value of that parameter which is a text string	10–94 2–95 7–95 4-96	6–289
PGS_CBP_body_inFOV	Given instrument parameters, returns a flag to indicate whether any of the user–selected major celestial bodies (sun, moon, etc.) are in the instrument field–of–view.	2–95 7–95	6–329
PGS_CBP_BrightStar_positions	Returns the position of all stars of magnitude > input magnitude, or the position of the stars designated by inputting star id's and setting the flag 'input_flag'.	TBD	6–336
PGS_CBP_Earth_CB_Vector	Computes the Earth centered inertial (ECI) frame vector from the Earth to the sun, moon, or planets at a given time, or range of time(s)	4–94, 10–94 7–95	6–318
PGS_CBP_Sat_CB_Vector	Computes the ECI vector from the spacecraft to the sun, moon, or planets at a given time or range of time(s)	4–94, 10–94 7–95	6–321
PGS_CBP_SolarTimeCoords	Computes local solar time, and right ascension and declination of the sun, for a given standard time and position on the surface of the Earth	4–94, 10–94 7–95	6–325
PGS_CSC_DayNight	Determines whether a given point on the Earth is in day, night or twilight, at a given time	10–94 7–95	6–429
PGS_CSC_Earthpt_FixedFOV	For a fixed field of view obtains the Coordinated Universal Time (UTC) time interval and the starting time that an Earth point is within the field–of–view, within a specified time window	4-96	6-390
PGS_CSC_Earthpt_FOV	For a field of view defined by a table of coordinates (accessed externally), and a known motion of the boresight vector as a function of time, obtains the Coordinated Universal Time (UTC) time interval and the starting time that an Earth point is within the field–of–view, within a specified time window	2–95 7–95	6–396
PGS_CSC_ECItoECR	Frame change tool	10-94 7/95	6-346
PGS_CSC_ECItoORB	Transforms a vector in the ECI Coordinate system to a vector in the Orbital Coordinate System	7–95	6–376
PGS_CSC_ECItoSC	Frame change tool	10–94	6–360
PGS_CSC_ECRtoECI	Frame change tool	10–94 7–95	6–350

Table 1–2. Toolkit Routine Listing (3 of 7)

Tool Name	Description	Date	Page
PGS_CSC_ECRtoGEO	Frame change tool	10–94 7–95	6–354
PGS_CSC_GEOtoECR	Frame change tool	10–94 7–95	6–357
PGS_CSC_GetFOV_Pixel	Computes the projection of (geolocates) the instrument field–of–view on the Earth, optionally, geolocates the center of each pixel in the footprint	4–94, 10–94 2–95 7–95	6–408
PGS_CSC_GreenwichHour	Returns the Greenwich Hour Angle of the vernal equinox, which is equal to Greenwich sidereal time, in the ECI frame, at a given time.	10–94	6–436
PGS_CSC_J2000toTOD	Transform from ECI J2000 to ECI True of Date	4-96	6-423
PGS_CSC_nutate2000	Transforms a vector under nutation from Celestial Coordinates of date in Barycentric Dynamical Time (TDB) to J2000 coordinates or from J2000 coordinates to Celestial Coordinates of date	7–95 4-96	6–419
PGS_CSC_ORBtoECI	Transforms vector in orbital coordinate system to vector in ECI coordinate system	7–95	6–380
PGS_CSC_ORBtoSC	Frame change tool	10–94 7–95	6–372
PGS_CSC_precs2000	Precesses a vector from Celestial Coordinates of date in Barycentric Dynamical Time (TDB) to J2000 coordinates or from J2000 coordinates to Celestial Coordinates of date in Barycentric Dynamical Time (TDB)	7–95	6–416
PGS_CSC_SCtoECI	Frame change tool	10–94	6–364
PGS_CSC_SCtoORB	Frame change tool	10–94 7–95	6–368
PGS_CSC_SpaceRefract	Estimate the refraction for a ray incident from space or a line of sight from space to the Earth's surface, based on the unrefracted zenith angle	7–95 4-96	6–403
PGS_CSC_SubSatPoint	Returns the position and velocity vector of the sub–satellite point ("pierce point"), or nadir of the satellite on the Earth's surface. Optionally returns the nadir vector also.	4–94, 10–94	6–384 2-434
PGS_CSC_TODtoJ2000	Transform from ECI True of Date to ECI J2000 Coordinates	4-96	6-426
PGS_CSC_ZenithAzimuth	Returns zenith and azimuth angles of spacecraft	10–94 2–95	6–440
PGS_CUC_Cons	Accesses constant values from a predetermined input file	2–95	6–460
PGS_CUC_Conv	Accesses conversion slope and intercept values, needed to convert between units	2–95	6–463
PGS_EPH_EphemAttit	Provides access to spacecraft ephemeris and attitude data for a given time range, interpolates the state vectors and spacecraft attitude to a specified time	4–94, 10–94 2–95 7–95 4-96	6–212
PGS_EPH_GetEphMet	returns the metadata associated with toolkit spacecraft ephemeris files	11-96	6-218
PGS_GCT_Init	Performs Geo–coordinate transformation initialization for the given projection with the given parameters	2–95 7–95	6–451
PGS_GCT_Proj	Performs Geo–coordinate transformations for the given projection in the forward and inverse directions	2–95 7–95	6–454
PGS_IO_Gen_Close	Close non–HDF file	4–94, 10–94	6–43
PGS_IO_Gen_CloseF	Close non–HDF file FORTRAN	10–94 7–95	6–45

Table 1–2. Toolkit Routine Listing (4 of 7)

Tool Name	Description	Date	Page
PGS_IO_Gen_Open	Open non–HDF file	4–94, 10–94 7–95	6–35
PGS_IO_Gen_OpenF	Open non–HDF file FORTRAN 77 & 90	10–94 2–95 7–95	6–39
PGS_IO_Gen_Temp_Delete	Permanently delete a temporary file	4–94, 10–94 2–95 7–95	6–104
PGS_IO_Gen_Temp_Open	Open temporary file	4–94, 10–94 2–95	6–95
PGS_IO_Gen_Temp_OpenF	Open temporary file FORTRAN 77 & 90	10–94 2–95	6–100
PGS_IO_L0_Close	Closes a virtual data set that was opened with a call to PGS_IO_L0_Open.	2–95 4-96	6–24
PGS_IO_L0_File_Sim	Creates a file of simulated Level 0 data	2–95 4-96	6–26
PGS_IO_L0_GetHeader	Gets the header and footer data for the currently open physical file	2–95 4-96	6–13
PGS_IO_L0_GetPacket	Gets a single packet from the specified Level 0 Virtual Data Set	2–95 4-96	6–19
PGS_IO_L0_Open	Open a Virtual Level 0 Data Set	2–95 4-96	6–6
PGS_IO_L0_SetStart	Sets the specified open virtual data set so that the next call to PGS_IO_L0_GetPacket will read the first packet at or after the specified time	2–95 4-96	6–11
PGS_MEM_Calloc	Allocates an array of arbitrarily sized elements, initializing them to zero, in memory	10–94 7–95	6–468
PGS_MEM_Free	Deallocates memory that was previously allocated	10–94 7–95	6–473
PGS_MEM_FreeAll	Deallocates all memory that was previously allocated within a process	10–94 7–95	6–474
PGS_MEM_Malloc	Allocates an arbitrary number of bytes in memory	10–94 7–95	6–466
PGS_MEM_Realloc	Reallocates the number of bytes requested	10–94 7–95	6–470
PGS_MEM_ShmAttach	Used by an executable to attach to an existing shared memory segment	10–94	6–199
PGS_MEM_ShmCreate	Used to create a shared memory segment	10–94	6–196
PGS_MEM_ShmDetach	Used to detach a shared memory segment from a process that attached it	10–94	6–201
PGS_MEM_ShmRead	FORTRAN Read from Shared Memory	4-96	6-203
PGS_MEM_ShmWrite	FORTRAN Write to Shared Memory	4-96	6-205
PGS_MEM_Zero	Initializes a memory block or structure to zero	10–94 7–95	6–472
PGS_MET_GetConfigData	Enables the user to get the values of Config data parameters held in the PC table	7–95 4-96	6–84

Table 1–2. Toolkit Routine Listing (5 of 7)

Tool Name	Description	Date	Page
PGS_MET_GetPCAttr	Retrieves parameter values from the PC table which are either located as HDF attributes on product files or in separate ASCII files	7–95 4-96	6–77
PGS_MET_GetSetAttr	Enables the user to get the values of metadata parameters which are already set by the initialization procedure	7–95 4-96	6–71
PGS_MET_Init	Initializes a metadata configuration file (MCF)	7–95 4-96	6–48
PGS_MET_Remove	Contains PGS_MET_Remove() which frees the memory held by the metadata configuration file (MCF) and data dictionary object description language (ODL) representations	7–95 4-96	6–93
PGS_MET_SetAttr	Enables the user to set the value of metadata parameters	7–95 4-96	6–61
PGS_MET_Write	Enables the user to write different groups of metadata to separate HDF attributes	7–95 4-96	6–89
PGS_PC_GenUniqueID	Used to generate a unique product identifier. May be attached to file metadata to facilitate tracking of production output	10–94 4-96	6–177
PGS_PC_GetConfigData	May be used to access run–time parameters in the PGE	10–94 4-96	6–179
PGS_PC_GetConfigDataCom	May be used to access run–time parameters at the shell level	2–95 4-96	6–159
PGS_PC_GetFileAttr	Used to retrieve the attribute string that contains the metadata for a Product file	10–94 4-96	6–185
PGS_PC_GetFileAttrCom	Used at the shell level to retrieve an attribute "stream" that contains the metadata for a Product file	2–95 4-96	6–161
PGS_PC_GetFileByAttr	Used to retrieve the specific instance of a product file that satisfies the search criteria, defined by a user–supplied method, applied to the metadata of each product file instance	10–94 4-96	6–188
PGS_PC_GetNumberOfFiles	May be used to query the number of file instances that are associated with a particular product file	10–94 4-96	6–182
PGS_PC_GetNumberOfFilesCom	May be used, at the shell level, to query the number of file instances that are associated with a particular product file	2–95 4-96	6–160
PGS_PC_GetReference	Used to obtain a physical file pathname from a logical identifier for a particular Product file	10–94 4-96	6–170
PGS_PC_GetReferenceCom	Used at the shell level to obtain a physical file pathname from a logical identifier for a particular Product file	2–95 4-96	6–156
PGS_PC_GetReferenceType	Tool may be used to ascertain the type of file reference which is associated with a logical identifier within the science software	7–95 4-96	6–173
PGS_PC_GetTempReferenceCom	Used at the shell level to obtain a physical file pathname from a logical identifier for a particular Temporary, or Intermediate file	2–95 7–95 4-96	6–164
PGS_PC_GetUniversalRef	Used to obtain a universal reference from a logical identifier	4-96	6–193
PGS_PC_InitCom	Used, prior to PGE execution, to establish a working environment for the SDP Toolkit	2–95 7–95 4-96	6–155
PGS_PC_Shell.sh	Provides an integrated environment for the SDP Toolkit and a PGE	2–95 7–95 4-96	6–152
PGS_PC_TempDeleteCom	Used at the shell level to delete the Temporary file currently associated with a particular logical identifier	2–95 4-96	6–167

Table 1–2. Toolkit Routine Listing (6 of 7)

Tool Name	Description	Date	Page
PGS_PC_TermCom	Used, following PGE termination, to cleanup the resources used by the SDP Toolkit	2–95 4-96	6–168
PGS_SMF_Begin	Signal SMF that function has started	4-96	6-144
PGS_SMF_CreateMsgTag	May be used to generate a unique message identifier	10–94 4-96	6–123
PGS_SMF_End	Signal SMF that function has ended	4-96	6-145
PGS_SMF_GenerateStatusReport	Used to add user–defined status reports to the Status Report Log file	10–94 4-96	6–127
PGS_SMF_GetActionByCode	Provide the means to retrieve an action string associated with a specific mnemonic code	10–94 4-96	6–121
PGS_SMF_GetInstrName	Used to retrieve the instrument name from a given error/status code	4–94, 10–94 4-96	6–125
PGS_SMF_GetMsg	Provide the means to retrieve a previously set message from the static buffer PGS_SMF_Set....	4–94, 10–94 4-96	6–120
PGS_SMF_GetMsgByCode	Provide the means to retrieve the message string corresponding to a specific mnemonic code	10–94 4-96	6–119
PGS_SMF_SendRuntimeData	Provide a means for the user to transmit a package of runtime data to the SCF in the event of an unhandled system exception	10–94 2–95 4-96	6–129
PGS_SMF_SetArithmeticTrap	Used to specify a signal handling function to perform in the event that an error arithmetic operation has occurred.	TBD	6–146
PGS_SMF_SetDynamicMsg	Provide the means to set a user–defined error/status message in response to the outcome of some segment of processing.	10–94 4-96	6–116
PGS_SMF_SetHDFMsg	Provide the means to retain the HDF–EOS error message as a result of an HDF–EOS error.	TBD	6–113
PGS_SMF_SetStaticMsg	Provide the means to set a predefined error/status message in response to the outcome of some segment of processing.	4–94, 10–94 4-96	6–114
PGS_SMF_SetUNIXMsg	Provides the means to retain UNIX error messages for later retrieval	4–94, 10–94 4-96	6–110
PGS_SMF_TestErrorLevel	Will return a Boolean value indicating whether or not the returned code has status level 'E'	10–94 4-96	6–133
PGS_SMF_TestFatalLevel	Will return a Boolean value indicating whether or not the returned code has status level 'F'	10–94 4-96	6–135
PGS_SMF_TestMessageLevel	Will return a Boolean value indicating whether or not the returned code has status level 'M'	10–94 4-96	6–136
PGS_SMF_TestNoticeLevel	Will return a Boolean value indicating whether or not the returned code has status level 'N'	10–94 4-96	6–140
PGS_SMF_TestStatusLevel	Will return a defined status level constant	4–94, 10–94 4-96	6–141
PGS_SMF_TestSuccessLevel	Will return a Boolean value indicating whether or not the returned code has status level 'S'	10–94 4-96	6–139
PGS_SMF_TestUserInfoLevel	Will return a Boolean value indicating whether or not the returned code has status level 'U'	10–94 4-96	6–138
PGS_SMF_TestWarningLevel	Will return a Boolean value indicating whether or not the returned code has status level 'W'	10–94 4-96	6–137

Table 1–2. Toolkit Routine Listing (7 of 7)

Tool Name	Description	Date	Page
PGS_TD_ASCIItime_AtoB	Converts binary time values to ASCII Code B time values of the form year_month_day_time_of_day in the Consultative Committee on space Data Systems (CCSDS) format	10–94	6–248
PGS_TD_ASCIItime_BtoA	Converts binary time values to ASCII Code A time values of the form year_month_day_time_of_day in the CCSDS format	10–94	6–250
PGS_TD_GPStoUTC	Converts to Coordinated Universal Time (UTC) time value from Global Positioning System (GPS) time by converting to internal time, adding the GPS_minus_UTC_leapseconds from the leapseconds file, and converting to GPS format following CCSDS ASCII standard A	10–94 7–95	6–254
PGS_TD_LeapSec	Find Leap second value	4-96	6-273
PGS_TD_SCtime_to_UTC	Converts spacecraft clock time to UTC for EOS platforms or for foreign spacecraft	4–94, 10–94	6–245
PGS_TD_TAItoGAST	Converts International Atomic Time (TAI) (toolkit internal time) to Greenwich apparent sidereal time (GAST) expressed as the hour angle of the true vernal equinox of date at the Greenwich meridian (in radians)	7–95	6–240
PGS_TD_TAIjdtoTAI	Converts TAI Julian date to time in TAI seconds since 12 AM UTC 1-1-1993.	4-96	6-238
PGS_TD_TAItoTAIjd	Converts time in TAI seconds since 12 AM UTC 1-1-1993 toTAI Julian date.	4-96	6-236
PGS_TD_TAItoUTC	Converts TAI time value to UTC time	4–94, 10–94	6–234
PGS_TD_TimeInterval	Computes the elapsed TAI time in seconds between any two epochs after January 1, 1958	10–94	6–262
PGS_TD_UTCtoGPS	Converts UTC time value to GPS time by converting to internal time, adding the GPS_minus_UTC_leapseconds from the leapseconds file, and converting to GPS format following CCSDS ASCII standard A	10–94 7–95	6–252
PGS_TD_UTCtoTAI	Converts UTC time to TAI time by first converting UTC to internal time and then adding the TAI_minus_UTC_leapseconds from the leapseconds file	4–94, 10–94	6–230
PGS_TD_UTCtoTDBjed	UTC to Barycentric Dynamical Time (TDB) time conversion	10–94	6–259
PGS_TD_UTCtoTDTjed	UTC to Terrestrial Dynamical Time (TDT) time conversion	10–94	6–256
PGS_TD_UTCtoUT1	Converts UTC to UT1 time	10–94	6–268
PGS_TD_UTCtoUT1jd	Converts UTC time in CCSDS ASCII Time Code to UT1 time as a Julian date	7–95	6–271
PGS_TD_UTCjdtoUTC	Converts UTC as a Julian date to UTC in CCSDS ASCII Time Code A format.	4-96	6-266
PGS_TD_UTCtoUTCjd	Converts UTC in CCSDS ASCII Time Code A format.to UTC as a Julian date.	4-96	6-264
PGS_TD_UTC_to_SCtime	Converts UTC to Spacecraft clock time for EOS standard of Foreign Spacecraft	10–94	6–242
smfcompile	Provides means to store messages in files that are accessed at runtime to get the message text.	4–94, 10–94 2–95	6–148

Note for Table 1–2: If more than one date is in the delivery column this indicates a re–delivery of that tool.

Table 1–3. Tool Changes for Release A Toolkit Delivery

Tool Name	Type of Change
PGS_CSC_ORBtoSC PGS_CSC_SCtoORB	removed tipped orbit correction for TRMM, since Toolkit will now receive orbit in J2000
PGS_EPH_EphemAttit	Revised ephemeris data files format to match Release A implementation and added quality flag support
PGS_EPH_GetEphMet	New tool
PGS_IO_L0_tools	Revised format to match June 1996 EDOS ICD
PGS_MET_tools	Can now read old format Metadata files
PGS_MET_tools	added support for multiple MCFs
PGS_MET_tools	Modify to conform to Rel A data server interface specifications
PGS_TD_NewLeap update_leapsec.sh	Now obtain leap second information from US Naval Observatory, instead of International Earth Rotation Service
Toolkit	all user support bugs fixed
Toolkit	now supports IRIX 6.2 operating system

1.5 Document Organization

The document is organized as follows:

Section 1 Introduction—Presents the scope and purpose of this document.

Section 2≠ Related Documentation—Provides a bibliography of reference documents for the science data production (SDP) Toolkits organized by parent and applicable documents.

Section 3≠ Toolkit Design Overview—Provides the philosophy and high level description of the Toolkit

Section 4≠ Toolkit Usage and Functionality—Describes the functionality to be provided in the SCF and follow–on SDP versions of the Toolkit.

Section 5≠ Toolkit Installation—Contains installation procedures for the machines for which Version 1 of the Toolkit has been certified.

Section 6≠ SDP Toolkit Specification—Contains calling sequences, description and usage examples for Toolkit routines.

Appendix A Assumptions

Appendix B Status Message File (SMF) Creation and Usage Guidelines

Appendix C Defining Process Control Files

Appendix D Ancillary Data Access Tools

Appendix E Example of Usage of Level 0 Access Tools

Appendix F Level 0 File Formats

Appendix G PGS_GCT Information Relating To Interface Specification

Appendix H PGS_CUC_Cons—Example Standard Constants File

Appendix I PGS_CUC_Conv—Input File Provided With the UdUnits Software

Appendix J Population of Granule Level Metadata using the SDP metadata tools

Appendix K POSIX Systems Calls Usage

Appendix L Ephemeris And Attitude File Formats

Appendix M Problem Identification List

Acronyms and Abbreviations

This page intentionally left blank.

2. Related Documentation

2.1 Parent Documents

The parent documents are the documents from which this SDP Toolkit Users Guide's scope and content are derived.

423-16-02≠	Goddard Space Flight Center, PGS Toolkit Requirements Specification for the ECS Project
423–41–01≠	Goddard Space Flight Center, EOSDIS Core System (ECS) Statement of Work, through CN-14, 4/25/95
423–41–02≠	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), through CH-06, 4/25/95
none≠	Goddard Space Flight Center, The PGS Toolkit Study Report, Version 1.9, 5/93

2.2 Applicable Documents

The following documents are referenced within this SDP Toolkit Users Guide, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume.

301–CD–002–003	System Implementation Plan for the ECS Project
194-WP-924-001	Level 0 Data Issues for the ECS Project, White Paper
445–TP–002–002≠	Theoretical Basis of the SDP Toolkit Geolocation Package for the ECS Project, Technical Paper
GSFC 50–003–04≠	Goddard Space Flight Center, EOSDIS Version 0 Data Product Implementation Guidelines (V1.0), 3/1/94
CCSDS 301.0–B-2≠	Consultative Committee for Space Data Systems (CCSDS) Recommendation for Space Data System Standards: Time Code Formats, 4/90
IEEE Std 1003.1≠	Institute of Electrical and Electronics Engineers; POSIX Part 1: System Application Program Interface (API)[CLanguage]
IEEE Std 1003.9≠	Institute of Electrical and Electronics Engineers; POSIX FORTRAN77 Language Interfaces, Part 1: Binding for System Application Program Interface [API]

none	Computer Science Corporation; Upper Atmosphere Research Satellite (UARS) Lessons Learned for EOS: Report 1—Design and Implementation (ending December 21, 1993); 5/92
none	University of Illinois/National Center for Supercomputing Applications; NCSA HDF Calling Interfaces and Utilities, Version 3.2; 3/93
none≠	University of Illinois; Getting Started With HDF, 1993 This is also available via anonymous file transfer protocol (ftp) from ftp.ncsa.uiuc.edu (141.142.20.50)
none≠	Wertz, J.R., Spacecraft Attitude Determination and Control, Reidel Publishing Co., 1984.

2.3 Information Documents

The following Internet link to a document/information, although not directly applicable, amplifies or clarifies the information presented in this document. This reference is not binding on this document. Internet links cannot be guaranteed for accuracy or currency.

194-815-SI4-001≠	SDP Toolkit Primer (current version available through WWW access: http://edhs1.gsfc.nasa.gov)

3. Toolkit Design Goals

The PGS Toolkit Requirements Specification served to create a specification for a compendium of tools that meet both ECS system requirements and the needs of the EOS science instrument data producers. The SDP Toolkit User's Guide represents the culmination of efforts to design tools that satisfy those criteria. In order to create that design, several broad features were devised to give the Toolkit a sense of continuity such that it may be considered a single tool with far–reaching capabilities.

3.1 Foundations

In order to ensure a high degree of portability and maintainability across a wide variety of computer platforms, the SDP Toolkit has been designed to conform to the POSIX.1 standard. With a few exceptions, this goal is met in the current implementation. Cases where vendors operating system or compiler implementation prevent strict adherence to the Portable Operating System Interface for Computer Environments (POSIX) standard will be minimized and worked as the standard matures. Additionally, some components of the Toolkit have been designed to incorporate proven COTS and other heritage software to provide functionality that is largely accepted by the user community and can be easily integrated into the Toolkit.

3.2 Nomenclature

The naming of the tools has been standardized to include two prefixes: one to denote its membership in the family of SDP tools and the other to indicate the general area of functionality covered by the tool. For example, a Toolkit routine that performs a time conversion will be prefixed with 'PGS_TD_'. The remaining portion of each name will be detailed enough to indicate the explicit functionality performed by the tool (e.g., "PGS_TD_UTCtoTAI").

3.3 Consistency

This feature was achieved by the creation of a method for setting and retrieving status values and status messages through the use of pre–defined error and status return codes and associated Toolkit routines. Some of these return codes are defined by the SDP system, but most of them will be defined by the users themselves to give them maximum control over their processing. All the SDP Toolkit routines have been designed to adhere to this status return mechanism; likewise, all the user developed software should incorporate this mechanism as well. The widespread use of this feature will serve to create software that is consistent in its approach to error handling and status reporting, is more readable, adheres to principles of modularity, and is easier to maintain.

3.4 Hierarchical Design

Finally, the SDP Toolkit was designed to provide different levels of service, depending on the requirements of the developer. Primarily, the Toolkit was designed to provide for all the necessary system–level interfaces. However, much of the Toolkit functionality incorporates value–added features to provide a higher level of service for the developers creating higher–order algorithms. In order to accomplish this, many of the Toolkit routines are designed to use the services of lower–level Toolkit routines. Some of the tools, such as the memory management routines are only required to have one or two levels of service; whereas others, like the ancillary data I/O routines, may have several different levels of service. It is important to note that whatever level of service is required, the Toolkit routine that provides that service will have been designed to use the services of a lower level Toolkit routine. This means that the applications programmer can use any of the Toolkit routines to develop their own level of service if there is not an explicit Toolkit routine that provides it.

3.5 Units

Generally, in the CBP, CSC, TD, and EPH sections of the Toolkit all physical quantities are in Standard International (SI) units, and all angles are in radians. The only exceptions to the use of SI units are a few cases where a "time" such as a Greenwich "time" that is really a measure of Earth rotation may be given in radians, or (for Julian Date) days, instead of seconds - please consult the individual tool entries on this issue. In some of the AA and GCT tools specialized units appropriate to the relevant data set may be used; please consult the individual entries.

The HDF subsetting functions use SI seconds.

Users who wish to work in units other than those in the Tools are urged to use great caution. For example, the tools that transform between the spacecraft reference frame and Earth-centered reference frames take into account the displacement of the spacecraft (in meters) from Earth center, when the user supplies other than a unit vector. (For unit vector input. only the direction is transformed). To use these transformations on vectors denominated, for example, in kilometers would result in nonsense.

3.6 Ranges and Limits of Validity; unit vectors

The following material applies to the CBP, CSC, TD, and EPH tools; the AA and GCT tools may follow different rules which are explained in the appropriate sections.

On output, all angles that represent a longitude or azimuth will be in the range $(-\pi , \pi)$, but on input the Toolkit is more forgiving: no limit is imposed, although most library trigonometric functions tend to lose accuracy when the argument is very large. By keeping the input range open this way we hope to simplify the task of the user who may, for example, want to transform from geodetic coordinates to rectangular coordinates a patch of the Earth's surface that bridges the longitude discontinuity at or near the international date line. There is no harm in entering values larger than π or less than $-\pi$ as derived, say, from offsets. Latitude is in the range $(-\pi/2 , \pi/2)$. Nadir and Zenith angles are in the range $(0 , \pi)$. Altitude can be arbitrary, but some tools return warnings or balk, with an error return, if a questionable altitude is detected; see the individual descriptions. Referring to Section 3.5, here again is a case where the inadvertent input of coordinates in kilometers (which the tools would take to be meters) could result in worthless output and a warning message, only, that the spacecraft was "subterranean."

In many cases, CSC group tools require a unit vector input. The varying accuracies of different platforms, and the danger of algorithmic error in case of inputting a non-unit vector where a unit vector is called for, dictated that the Toolkit simply make a normalized copy of the vector for internal use anyway. Thus, users need not, in practice, normalize "unit vectors" supplied to our CSC functions. On output, when a unit vector is promised, however, a unit vector will be produced.

Certain time streams have limited range by the nature of their definition, as explained in the TD section. Generally, the broadest range of times is encompassed by the Julian Date time streams, but Toolkit time, secTAI93, will yield microsecond precision from 1960 to 2135 AD on 32 bit platforms.

The algorithms have been carefully chosen to preserve machine word precision where possible, but a few transformations are subject to some limitations explained in the individual entries. For example, as noted by Galileo and Copernicus, the apparent velocity of the Sun or a planet as viewed in a reference frame rotating with the Earth is absurdly large; therefore we do not calculate such velocities past the mean distance to the Moon.

3.7 Aging and Maturation Effects

Any tools, such as geolocation functions, that depend on a precise knowledge of Earth rotation, yield answers that depend, ultimately on measurement; Earth rotation cannot be predicted well enough to allow ultra-precise real time geolocation! Therefore, along with leap seconds data, the Toolkit imports, weekly, data files on Earth rotation from the U.S. Naval Observatory. Users who want precise Earth position can get it within a few centimeters, but they have to wait a week till the latest file is in! Users content with meter accuracy can process in real time, but if they reprocess later, their geolocation answers may change by several centimeters, or even a meter. For more details, see the SDP Toolkit FAQ on the EDHS home page on the World Wide Web, under Validation of the SDP Toolkit.

This page intentionally left blank.

4. Toolkit Usage, Functionality and Future Direction

4.1 Introduction

This User's Guide addresses the usage of the SCF version of the SDP Toolkit. As explained in *The PGS Toolkit Requirements Specification*, the purpose of the SCF development environment and Toolkit is (1) to provide development Toolkit functions that emulate the production Toolkit functions, (2) to provide a development environment that emulates the production environment to support development and test, (3) make both functions and environment easy to use, and (4) most importantly, allow for a smooth transition of science software from the SCF to the production environment, during the integration and test phase.

The ECS science software developer will use the Toolkit to access the production environment and services, or their emulation. The Toolkit routines are divided into two classes:

a. Mandatory:

In order to access production services such as scheduling and messaging services in a consistent way, to avoid duplication of science software development effort, and to assure portability across computing platforms, usage of a subset of the Toolkit functions is required. These include functions that deal with file I/O, error message transactions, process control, ancillary data access, spacecraft ephemeris and attitude, and time and date transformations. The use of these tools will be enforced through automatic checks at integration time at the DAACs.

b. Optional:

Other useful functions required by developers, such as those involving celestial body positions, coordinate transformations, math libraries, physical constants, and graphics support, will be provided by the Toolkit. The use of these services is optional, but is encouraged. Science software developers who use alternative solutions will be required to deliver the source code (Portable Operating System Interface for Computer Environments (POSIX) compliant) for the replacement services as part of the algorithm delivery. Prohibited and allowed system calls are the subject of Appendix K on POSIX.

The Toolkit will serve to insulate science software from the Science Data Processing (SDP) software, and to provide a development environment that emulates critical SDP-functions. In most cases, a complete simulation of the DAAC SDP System will not be required. The Toolkit will help ensure code portability as the algorithm is ported from development hardware, through the DAAC system, and through potential hardware changes as the ECS matures. To do so effectively, the Toolkit will provide for limited access and control to system level resources, including processes, shared memory, and I/O capabilities. Where control of such resources is necessary (e.g., shared memory allocation), the Toolkit will provide a set of routines through which the application must obtain those services. This partitioning and layering of operating system services allows the

Toolkit to work on behalf of the Data Processing subsystem in allocating, de–allocating, and making use of system–wide shared resources. The Toolkit will also serve to minimize code development by providing common functionality required across the ECS community, such as geolocation.

It is essential to understand the concepts that distinguish the SCF development environment from the production environment. While the science software and interface to the SDP Toolkit are preserved in both environments, there are slightly different implementations and behavior in the Toolkit functions and peripheral components (e.g., shell level development external to the product generation executive (PGE) and testing tools). As far as the calling sequences themselves go, these differences are transparent to the science software developer, i.e., the calling sequences in the SCF and production environment versions are identical. Some setup of the underlying environment will be necessary at the SCF, as explained in Section 4.2 below. This setup should not affect the code itself.

4.2 SCF Development Environment

4.2.1 Introduction

This User's Guide describes tools that were designed to function in the production environment. For this reason, certain assumptions were made during their design process which will affect the operation of these tools in the SCF environment. It is the primary purpose of this section to identify those areas where extra measures will need to be taken, on the part of the SCF developers, to compensate for the differences in the two environments. To assist with this effort, utilities that are being developed to support ECS internal testing will be made available to SCF developers after they are developed and tested. These utilities are expected to prove useful to Product Generation Executive (PGE) script development as well as to the integration and testing processes. Aside from supplying the production environment emulation services necessary to fully utilize the SDP Toolkit, these utilities will also provide an integrated environment to facilitate the specification and execution of test scenarios. Production environment emulation utilities will evolve over time as the architecture and system design of the ECS progress.

It is also the intent of this section to impart to the SCF developers our view of how science software development should be undertaken at the SCFs where the Toolkit is concerned. The intent here is merely to present our views and not to impose guidelines on the actual development process. If a future implementation of ECS, for example, allows for standard product production at the SCFs, usage of the Tools and utilities presented in this document should not impede but aid algorithm development.

4.2.2 File Management

In the production environment, product files coming from the system archive will be designated with an ECS–defined identifier. In order for science software to ingest these files, a scheme for translating internal software identifiers into actual physical identifiers has been established (Section 6.2.3). The same holds true for the SCF environment since the same I/O tools will be used to access these files from within the science software. The main difference being that in the production environment, these filename references are resolved when a PGE is queued for execution. Since the production environment will not be part of the SCF environment, a mechanism was devised to substitute for this functionality. This mechanism, known as a process control file (PCF), involves the creation of an external mapping of logical identifiers to physical file names according to the specifications for such a mapping.[1]

Some other notes regarding files concern the support for a one–to–many, logical–to–physical relationship among Product Input files. While this functionality is supported by the Toolkit, there are several guidelines that must be observed when defining these associations through the PCF mechanism. The first of these requires that files can only have more than one *instance* if they are entered into the section of the PCF labeled PRODUCT INPUT FILES. Since the logical identifier is static for files of this type, an *instance* number is required by the Toolkit, when references are made, to disambiguate among several files in the group. In order to ascertain the number of *instances* associated with a logical identifier, you must invoke the Toolkit function that provides this information (Section 6.2.3.2). Second, the order in which associated Product Input files is retrieved, using a sequentially increasing *instance* number, is the same order in which they are presented in the PCF, e.g., an *instance* number of 3 indicates the third associated Product Input file defined in the PCF. <u>Third, associated Product Input Files are those which possess the same logical identifier and appear in succession in the PCF.</u> Lastly, the *instance* number is **NOT** directly related to the *sequence* number that appears at the end of the Record Field in the PCF for each Product Input file (Appendix C).

Until more is known about the ability to request that Product Input files be staged (loaded to disk & updated in the PCF) in a specific order, we recommend that you **NOT** anticipate that such an ordering will exist in the production environment. Rather, always examine the file attributes (metadata) to ascertain the specifics about the Product Input file before referencing it.

The utility we propose will provide functionality for a user to easily associate known physical file names with the logical file identifiers used when invoking SDP Toolkit functionality. These associations will be created as part of generating a test scenario just prior to PGE execution and may be modified for subsequent test runs. Prior to creating any test scenarios, users of this utility will perform an initial setup of a PCF for each PGE. This procedure includes the entering of required file names (i.e., physical names of test data files); the sources (hostname:pathname) of these files; the locations of any translation software required to be activated for these files; and the preferred method of file access (e.g., mount, ftp) at the SCF. This functionality will be performed by the Processing system in the production environment.

It is our intent that this utility may be used to create some basic test scenarios, principally through the unification of logical and physical file identifiers, and to have those scenarios played–out by activating execution of the PGE for which the scenarios were devised. In so doing, all requested files will be dispatched to the runtime host, if so dictated, where any necessary translation will be performed before actual execution of the PGE occurs.

---

[1] The specification for file name mapping is defined in Appendix C - Defining Process Control Files

4.2.3 Runtime Configuration

To support a wide range of testing scenarios, some runtime parameters may be required to modify the behavior of the PGE under certain conditions. The SDP Toolkit contains the routines necessary to access the values of these parameters during runtime, provided that an external mapping of logical identifiers to actual values has been performed according to the specifications for this type of mapping.[2]

In the production environment, control of these parameters will most likely occur through a client interface that constructs a production request; the parameter changes resulting from activation of this mechanism will override the default mapping maintained in the production environment. In order to provide developers with some means of modifying the values of these parameters as part of scenario testing, an additional utility will be designed to perform this function. It is our intent to integrate this utility with the one outlined previously to create a more comprehensive scenario testing tool.

4.2.4 PGE Script Development

It is our view that PGE scripts will be created by SCF science software developers to build the logical framework around the executables that will produce the science products. It is also our understanding that the same Product Generation Executive (PGE) script will be delivered to the DAAC integration & test (I&T) team with little or no modifications required. In order to achieve this, the actual script should ideally be developed using a POSIX.2 conforming shell language. If at the time of development such a shell is not supported for all the approved platforms, development may proceed by using the standard Bourne shell (or other shell language approved by the ECS Project) on those platforms lacking a POSIX.2 implementation.

The actual PGE script as initiated in the production environment will not take arguments from the command line. Instead, script calls to command versions[3] of some 'Process Control' tools (see Section 6.2.3) will provide for the retrieval of pertinent runtime information. Likewise, the routine versions of the same tools should be used to obtain runtime information from within the executables, rather than passing this information through the shell interface. This allows for easier configuration of executables within the PGE script should modifications be required at some point in the future. This scheme is possible since the executable interfaces, files and runtime parameters, are defined and maintained external to the PGE script in the production environment. It is to the SCF developer's benefit to adhere to this convention wherever possible, to ensure portability of software into the production environment.

To support the startup and housekeeping needs of the SDP Toolkit; a Toolkit shell command has been developed which performs the necessary initialization and termination procedures. This shell command accepts a PGE script as input, assuring that execution of the PGE occurs between the

---

[2] Refer to Appendix C.

[3≠] Command versions of certain Process Control tools were not included in this version of the User's Guide, but will be defined in a future release of the User's Guide.

initialization and termination phases of the Toolkit. This shell command is similar to that which will be run in the production environment to guarantee the proper activation and deactivation of the Toolkit. It is recommended that SCF developers utilize this tool when conducting their testing.

When testing for the exit status of an executable within the PGE, only two values should exist[4]: (0) for success and (1) for failure. This will require the executable developers to invoke the library exit call with the appropriate value as the final statement in their software. The same holds true for the exit status of the PGE with the exception that the shell command 'exit' is invoked instead.

The toolkit will support the following script languages: Bourne shell, C shell, Korn shell, POSIX and the Perl language.

4.2.5 Scheduling and Execution of PGEs

As was previously stated, scheduling or queuing of PGEs via the data production processing subsystem will not be part of the SCF. However, developers should be able to use the emulation utilities mentioned earlier to generate scenario scripts for different PGEs. With each scenario script tailored to execute a single PGE for specific set of conditions, a superscript that activates several scenario scripts could be used to perform the execution of multiple PGEs.

While long term planning is not an inherent feature of this utility, UNIX cron facilities may be used within such a script to achieve some short term planning.

4.2.6 Error/Status Message Creation and Use

The 'smfcompile' utility provided in the SDP Toolkit (see Section 6.2.2) contains all the required functionality for defining and maintaining error and status codes, user messages and associated action messages. This tool, while only used in the SCF development environment, will fully support the suite of 'Error and Status Reporting' tools at both the SCF and the production environment.

Designed to support modular program development, this utility allows for separating the task of defining status codes and messages from the actual software development task. In fact, the process of defining these codes and messages may even be performed in the design stage, only later to be referenced during software development. This is an especially useful arrangement if action messages are to be incorporated into the status codes. This way, someone other than the programmer can decide the action that needs to be taken when a certain error, or status condition occurs.

---

[4]≠ Current shells only support status return values from 0 to 255. In order for the Toolkit to support error/status returns from executables and PGEs, an established list of system-wide error values will need to be defined. Caution: the 'csh' only supports signed return values, so care should be exercised if large values (> 127) are interpreted by the PGE between executables.

While we do not endorse the creation of a separate Status Message File (SMF) for each routine/function, etc., we do recommend that SMF file creation follow the logical partitioning of software modules. So for a related set of routines, or even a small program, there might only be one SMF that defines the status codes and messages returned by those routines.

The format for defining a status code mnemonic is intentionally free–form to allow the developer to create and reference status codes that convey some meaning when writing and visually inspecting the code.

4.2.7 Error/Status Log Monitoring

In the production environment, an error/status log file will be opened just before execution of the PGE. This will be accomplished through an 'Initialization' command invoked by the production environment. This tool and its associated 'Termination' command, were delivered with the Toolkit 4 release of the SDP Toolkit. Developers can insert these tools at the beginning and end respectively of a superscript that encapsulates their PGE script. However, it would be preferable to use the Toolkit Shell command, since it already calls these commands and provides for the encapsulation of a PGE script. If the emulation utility is used, the scenario scripts that it generates will automatically incorporate these tools.

The actual log file name will most likely be influenced by system parameters in the production environment. The easiest mechanism for accomplishing this in the SCF environment will be through the assignment of some file name to the appropriate Record Field, in the process control file (PCF), for each status log. The emulation utility may allow for the log file to be defined through the file management services.

Through the emulation utility, the actual name of the log file could be conveyed to the user on the host platform's console, or through some other convenient means. The user will most likely initiate a scrolling output of the log file to a terminal window, to monitor the progress of PGE execution.

The 'Termination' command mentioned earlier will be responsible for closing the log file and dispatching it to some pre–defined location as specified by a Runtime Parameter in the PCF.

4.2.8 Parallel Processing Issues

While the majority of the software to be designed at the various SCF locations will be sequential in nature, due to its direct dependency on data, some portion of that software lends itself to being processed in a parallel fashion. This is especially true of those processes that share a common set of input data, but which have no interdependencies themselves.

Unfortunately, the system architecture that would define the ability to execute portions of a PGE on non–host platforms (i.e., a massively parallel machine) in the production environment has not yet been determined. Until such an architecture is defined, if at all, developers will only be able to test concurrent execution of executables on a single host. If a requirement for this type of processing is derived, the toolkit will be configured to work in that environment.

4.2.9 Configuration Management

The importance of having a robust configuration management (CM) tool for a project of this size cannot be overstated. From SDP Toolkit development to science software development and integration, the use of this tool will control the version of software to support the continuous development and execution of production software.

After careful evaluation of several CM products, the ClearCase tool from Atria Software was chosen to support the internal software development, during site CM and maintenance and operations (M&O) CM requirements analysis. It is recommended that compatibility and interoperability benefits be explored.

4.2.10 Distributed Computing Environment (DCE) Issues

With the advent of distributed computing, an ever increasing amount of single process execution will be performed across multiple machines instead of the more typical scenario of many processes running on one machine. While this technology may someday help to improve the efficiency of your process, and at the same time take advantage of underutilized processors, the constraints that it places on the ECS system architecture preclude the use of certain Distributed Computing Environment (DCE) features. Among them is the use of remote procedure calls (RPC's). The creation of RPC's to perform some segment of processing for a science algorithm is such a customized task that its interface cannot be generalized into some extended SDP Toolkit functionality. Since it is the Toolkit's charter to isolate the science software from the system architecture, the SDP Toolkit's inability to mask this feature prohibits its direct usage in the actual production software. For this reason, the direct use of RPC’s will not be allowed in the algorithm software developed by the instrument teams.

We note that an interface that makes RPC’s indirectly available to science users through a client interface is being considered in revisions of the ECS architecture. This interface may become an extension of the Toolkit and will allow the algorithmic access of data through parameter–based searches. The details and limitations of this interface are not available at the time of this document.

4.3 Test and Simulation Data Access

The Toolkit provides tools to access all external data files required for science processing development and execution. There are tools to provide the read functions to all data types: L0, ancillary data, calibration coefficient files, standard products, etc.

Clearly test data must be accessed through the tool with the same toolkit interface as in the production environment. In general the Toolkit will aim to provide low level “write” functions to match the “read” functions so that the users may develop their own test data sets to the format required. Although there are currently no requirements that the Toolkit supply these new "write" tools, it is expected that they will be required for adequate testing within the production environment. In certain cases, such as platform orbit and attitude simulation, the Toolkit may provide specially prepared test data sets.

For example, the L0 data write tool will provide a function to write data into a file formatted as the packet based structure expected from EOSDIS Data and Operations System (EDOS). In this example the “write” routine would require that the science data is provided by the user so that the test data set may be tailored to the user needs.

For the case of dynamic external auxiliary data (e.g., Special Sensor for Microwave Imaging (SSM/I) water vapor data) software may be provided to preprocess external data into any internal format used in the production environment, so that consistent data sets for testing may be developed by the user as required.

In the current implementation, EOS AM, EOS PM and Tropical Rainfall Measuring Mission (TRMM) orbit and attitude simulation are supplied with the Toolkit. A packetizing tool Level 0 instrument science data simulation (which can be used in conjunction with the orbit simulator) is under development. A digital chart of the world (DCW) data base and a celestial body ephemerides are also provided with the current delivery.

4.4 Language Bindings and Advanced FORTRAN Considerations

The calling sequences in this document are provided in the C language with FORTRAN calling sequences provided in addition for most tools.

a.≠ The SDP Toolkit is designed in C, with most of the FORTRAN interface provided via inter–language bindings. In cases where there is no obvious relationship between FORTRAN and C calls, i.e., C pointers and structures, bindings will have to be done carefully so as not to cause processing impairment. Note that there are no such tools in the current implementation.

b.≠ The question of computing speed has a strong effect on the design of FORTRAN tools. Some tools, such as Level 0 I/O tools, need to be as fast as possible—the extra layer of binding from C to FORTRAN may slow the processing to the point that the tool is unusable. Therefore a subset of the SDP Toolkit may be designed as FORTRAN—only; i.e., not bound to C, for this reason. The user interface will not change, however.

c.≠ FORTRAN77 is currently the highest level of FORTRAN that has a POSIX standard. However, many features of FORTRAN90 that are not present in FORTRAN77 are desired for inclusion in the Toolkit. These FORTRAN90 features include pointers and structures. This may mean that there will be two sets of FORTRAN calling sequences, one for 77 and one for 90. There are no FORTRAN90 only constructs in the current implementation. The tools compile in both FORTRAN77 and FORTRAN90 (NAG compiler).

d.≠ C++ is under consideration as a supported language at the time of Version 1. Barring an unlikely complete object oriented redesign of the Toolkit, the C routines included in the Toolkit will not be affected and no additional C++ bindings will be required. The issue, however, is under investigation.

e.≠ The only Ada support offered by the Toolkit is in the generation of Status Message Files by the 'smfcompile' utility.

5. Toolkit Installation and Maintenance

5.1 Installation Procedures

5.1.1 Release A Toolkit Release Notes

5.1.1.1 Multiple Architecture Support Added, Directory Structure Revisions

The toolkit now has the option of being installed with simultaneous support for multiple architectures. This means that it is no longer necessary to install a separate copy of the toolkit for each host architecture to be supported. Instead, a single copy of the toolkit, installed on a file server in a networked environment, may serve multiple hosts of different architecture types.

Running concurrent tasks on the toolkit is possible, but it requires that each process be configured so that all output files, including toolkit log files, are written to a separate area to avoid collisions. This is done by using a private customized Process Control File (PCF) for each concurrent task. Please refer to Appendix C for more information. Note that any such PCF MUST contain all of the entries in the master template PCf for prpoer toolkit functioning.

The directory structure of the toolkit has been revised to allow multiple architecture support. Subdirectories of the toolkit home directory are now as follows:

bin	binary and script executables	Note 1
database	data resource files used by the toolkit	Note 1
doc	documentation	
include	header files	
lib	the toolkit library	Note 1
message	message files used by the error/status tools	
obj	object files used to build the toolkit library	Note 1
runtime	runtime files	Note 2
src	source code	
test	test area	

Note 1:

The directories bin, database, lib and obj all contain architecture-specific files residing in subdirectories named for the architecture. One such subdirectory will be created for each run of the installation script on a given architecture. Toolkit environment variables are set by the environment scripts to automatically map to the appropriate directories.

Note 2:

The directory runtime contains data files shared by all architectures. In addition, it contains one subdirectory for the each of the supported architectures. These subdirectories are for architecture-specific runtime files. Currently the only file distributed in these subdirectories the default Process Control File (PCF) PCF.relA, which contains architecture-specific pathnames.

5.1.1.2 Support for the SGI Challenge

The toolkit now fully supports the SGI Challenge platform. The SGI Challenge can compile and run binaries in three different formats. The toolkit treats each of these formats as a separate architecture. The table below gives the formats:

binary format	compiler flag	toolkit name
old-style 32 bit	-32	sgi
new-style 32 bit	-n32	sgi32
64 bit	-64	sgi64

The old-style 32 bit format is backwards-compatible with 32-bit SGI platforms. The other formats run only on the Challenge.

5.1.1.3 HDF Integration

The toolkit installation procedures include a section that covers the installation of the National Center for Supercomputer Applications (NCSA) HDF file access package. HDF has been adopted as the baseline standard for EOSDIS Core System development of data formats for product generation, archival, ingest, and distribution capabilities.

Currently, HDF is only needed in order to build and use the Metadata (MET) and Ancillary Access (AA) tools. If you do not plan to use these tools, the HDF installation section may be skipped. In future releases, we expect greater integration of the toolkit with HDF.

An installation script for HDF is included as part of the main SDP Toolkit distribution. It is provided to simplify the installation of HDF as much as possible, greatly reducing the number of steps in NCSA's own installation procedure. Starting with Release A, the toolkit uses HDF 4.0r2. The HDF distribution itself is located in a compressed tar file, called HDF4.0r2.tar.Z which must be downloaded separately.

With a full installation, HDF requires approximately 30 Mb of disk space, after the installation files are cleaned up. It may be installed in any location; i.e., it does not have to be stored under the SDP Toolkit home directory. The disk partition where HDF is installed should have about 60 Mb of free space.

5.1.1.4 HDF-EOS Integration

The toolkit installation procedures now include a section which covers the installation of HDF-EOS, a standalone package that may be used in conjunction with the toolkit. It implements the EOS standard methods for accessing HDF format files. Three interfaces are provided: Point, Swath and Grid. Please refer to the HDF-EOS User's Guide for more information. The distribution file for HDF-EOS is available from the same ftp server where the toolkit distribution files are located.

The toolkit HDF-EOS installation is only available if the toolkit is built with HDF support. It handles the details of unpacking the distribution file, setting HDF dependencies, and running the HDF-EOS installation script.

HDF-EOS may also be installed manually, either before or after the toolkit is installed. HDF must be installed before installing HDF-EOS.

5.1.2 To Install the SDP (formerly PGS) Toolkit from a Disk–Based Tar File

5.1.2.1 Preliminary

If HDF is to be installed at this time (recommended), you must first download the HDF distribution file HDF4.0r2.tar.Z before proceeding. It may be loaded into any directory on your system, i.e. it need not reside in the SDP Toolkit home directory.

Important HDF Note:

The toolkit-supplied HDF installation script contains various platform-specific patches and bug fixes that allow HDF to be successfully installed on all platforms supported by the toolkit. In most cases, both the libraries and utilities are built. Due to various compatibility problems, only the libraries are currently supported on the Cray/UNICOS and SGI/IRIX64 (SGI Challenge) platforms.

Because of these factors, we strongly recommend that even if you already have HDF 4.0r2 installed, you RE-INSTALL HDF AT THIS TIME, using the toolkit-supplied HDF installation script.

Historical Note:

Please note the acronym PGS (Product Generation System) is used throughout the toolkit software in place of SDP. This is for historical reasons: the name was changed as of Release 3 of the toolkit. We regret any confusion this may cause.

5.1.2.2 Unpacking the Distribution File

1. Select a location for the SDP Toolkit directory tree. It should be on a disk partition with at least 80 Mb of free space. If you plan to install HDF in the same partition, you will need at least 110 Mb of free space.If you plan to install support for multiple architectures, you will need about 20 Mb Toolkit space + 30 Mb HDF space for each additional architecture supported.

   Multiple Architecture Support Note
   As previously mentioned, it is now possible to build the toolkit with support for multiple architectures. The distribution file need only be unpacked once, to support all architectures. If the toolkit is to be built with multiple architecture support, the area chosen to unpack the

distribution should be on a network file system accessible from all hosts to be supported. (Please note that the SGI Challenge supports three different architectures. So, if building a multiple architecture installation to support the Challenge only, the filesystem need not be accessible across the network.)

2. Copy the file SDPTK5.1.1v1.00.tar.Z to the target directory by typing the command:

    cp SDPTK5.1.1v1.00.tar.Z <target-dir>

   where <target-dir> is the full pathname of your target directory.

3. Set your default directory to the target directory by typing the command:

    cd <target-dir>

4. Uncompress this file and extract the contents by typing the command:

    zcat SDPTK5.1.1v1.00.tar.Z | tar xvf -

   This will create a subdirectory of the current directory called TOOLKIT. This is the top-level toolkit directory, which contains the full toolkit directory structure.

5.1.2.3 Starting the Installation Procedure

1. Set your default directory to the top-level toolkit directory by typing the command:

    cd TOOLKIT

   Multiple Architecture Support Note:
   The toolkit installation script must be run once for each of the architectures to be supported. To do this, simply login to the desired host and set your directory to the top-level toolkit directory: <target-dir>/TOOLKIT. Then, proceed to run the installation script, starting at Step 2, below. The installation runs MUST be done ONE AT A TIME. Attempting to run concurrent installation procedures will cause errors.

2. Determine options for the toolkit installation script.

   Before running the toolkit installation script, you must determine the command line options appropriate for your site. These options are refered to in this section as <install-options>.

   These options tell the installation script such things as whether to build for SCF or DAAC, and whether to build for FORTRAN-90 compatibility, (FORTRAN-77 is the default). The table below gives the basic site options. Other options follow.

Site	FORTRAN	<install-options>
SCF	FORTRAN-77	(none)
SCF	FORTRAN-90	-f90
DAAC	FORTRAN-77	-daac
DAAC	FORTRAN-90	-daac -f90

Please refer to part 1 of the Notes section, below, for information about platforms that currently support FORTRAN-90. When doing a FORTRAN-90 installation, the use of -fc_path option, (see below), is highly recommended.

If you wish to save the output of the installation run in a log file (RECOMMENDED), add the following to <install-options>:

-log <log-file>

Where <log-file> is the name of the log file.

To ensure that the proper C compiler is found by the script, you may use the -cc_path option to specify its location. To do this, add the following to <install-options>:

-cc_path <C-compiler-path>

Where <C-compiler-path> is the directory where the desired C compiler is located. This option should not be needed at most sites.

To ensure that the proper FORTRAN compiler is found by the script, you may use the -fc_path option to specify its location. To do this, add the following to <install-options>:

-fc_path <FORTRAN-compiler-path>

Where <FORTRAN-compiler-path> is the directory where the desired FORTRAN compiler is located. This is particularly advisable when using FORTRAN-90.

NAG FORTRAN-90 Note:

If using a NAG FORTRAN-90 compiler to build the toolkit library,add the -nag option to <install-options>, after the -f90 and-fc_path options. This will allow the toolkit to generate the proper C to FORTRAN bindings. This option should not be used when building the toolkit on the SGI Challenge. See the note, below.

SGI Challenge Multiple Architectures Note:

On the SGI Challenge, the default is to build the toolkit in old-style 32-bit mode. The following table gives the option to specify the appropriate achitecture to be built:

binary format	architecture	<install-options>
old-style 32 bit	sgi	(default)
new-style 32 bit	sgi32	-sgi32
64 bit	sgi64	-sgi64

SGI and SGI Challenge FORTRAN-90 Note:

On SGI and SGI Challenge platforms running IRIX 6.1 and earlier, the type of FORTRAN-90 compiler is automatically determined by the script. On the 32-bit SGI platform, the NAG compiler is used. On the 64-bit SGI Challenge platform, the compiler chosen depends on the binary architecture type selected.

The script will override the setting of the -NAG flag, if specified, because only the combination listed below will build properly. The following table shows which compiler is used for each architecture:

binary format	architecture	f90
old-style 32 bit	sgi	NAG
new-style 32 bit	sgi32	SGI
64 bit	sgi64	SGI

On SGI and SGI Challenge platforms running IRIX 6.2 and later, the script will set up to use the SGI f90 compiler unless the -NAG option was specified.

IMPORTANT NOTE: In order to build the old-style 32-bit toolkit with the SGI f90 compiler (IRIX 6.2 and later only), the compiler MUST be version 7 or later. Otherwise, the NAG f90 compiler should be used.

5.1.2.4 Run the Toolkit Installation Script

Please note that the installation script for this release of the toolkit requires user interaction. Because of this, it should NOT be run as a background task. The new installation script, bin/INSTALL, is actually a front end for two other scripts: bin/INSTALL-HDF and bin/INSTALL-Toolkit. Each of these scripts may be run with the -h option to display a usage message. Under normal circumstances it should never be necessary to run either bin/INSTALL-HDF or bin/INSTALL-Toolkit standalone.

To run the script, type the command:

bin/INSTALL <install-options>
where <install-options> is the list of options determined in the previous step.

The installation script will then run. It will output various startup messages, beginning with:

Toolkit Installation starting at <date/time>

The script will then output a message discussing the HDF requirement,after which it issues a prompt which gives you an opportunity to quit.

Continue installation [yes] ?

To continue the installation, press return.

HDF Installation Section

1. The script prompts with:

   Is HDF4.0r2 installed at your site [no] ?

   If HDF is not installed, hit return and proceed to step 3, below.

2. If you already have the correct version of HDF installed, you may type 'y' and hit return. In this case, the script will ask where HDF is installed:

   Pathname where directory HDF4.0r2 is located [<default>] ?

   Type in the full pathname and hit return. The script will check to make sure that HDF is really installed there. Please proceed to the toolkit Installation Section, below.

3. The script prompts with:

   Do you wish to install HDF4.0r2 now [yes] ?

   Hit return to continue.

4. The script responds with:

   Running the HDF Installation Script ...

   It may also output a few informational messages, depending on the installation options selected.

5. By default, the script looks for the distribution file in your current and parent directories. If the file is found in either of these locations, the script will continue to the next step. Otherwise, it will prompt with:

   Pathname where HDF4.0r2.tar.Z is located ?

   Please enter the correct location and hit return.

6. The script then asks where the HDF directory will be created. The default is <toolkit-home-directory>/hdf/$BRAND, where $BRAND is the toolkit architecture being built, given by the table in Note 2 of the NOTES section, below.

   Pathname where directory 'HDF4.0r2' will be created [<default>] ?

   If you want HDF installed elsewhere, please enter the pathname at the prompt. Otherwise, simply hit return to continue.

   Multiple Architecture Support Note:

   A copy of the HDF installation must be built for each of the architectures to be supported by this toolkit installation. We therefore recommend using the default HDF directory, suggested by the installation procedure, as it helps keep track of which architecture was used to build HDF.

7. The script asks you to verify the information entered, prompting with:

   Continue [yes] ?

   Hit return to continue. The contents of the distribution file are then extracted into the specified location, and the installation procedure is run.

8. This completes the interactive portion of the HDF installation. When the HDF section is complete, it outputs the message:

   HDF installation ending at: <date/time>

HDF-EOS Installation Section

1. The script prompts with:

   Do you wish to install HDF-EOS1.5v1.00 now [yes] ?

   Hit return to continue.

2. The script responds with:

   Installing HDF-EOS ...

   It may also output a few informational messages, depending on the installation options selected.

3. By default, the script looks for the distribution file in your current and parent directories. If the file is found in either of these locations, the script will continue to the next step. Otherwise, it will prompt with:

   Pathname where HDF-EOS1.5v1.00.tar.Z is located ?

   Please enter the correct location and hit return.

4. The script then asks where the HDF directory will be created. The default is <toolkit-home-directory>.

   Pathname where directory 'hdfeos' will be created [<default>] ?

   If you want HDF-EOS installed elsewhere, please enter the pathname at the prompt. Otherwise, simply hit return to continue. If installing for an additional architecture, (refer to the Multiple Architecture Support Note in Step 1 of "Starting the installation procedure"), use the same directory as for the first instance of HDF-EOS - a single copy will support multiple architectures.

5A. Single-Architecture Installation

   If this is a single-architecture installation, or the first platform of a multiple-architecture installation, do this step. Otherwise proceed to step 5B.

   The script asks you to verify the information entered, prompting with:

   Continue [yes] ?

   Hit return to continue. The contents of the distribution file are then extracted into the specified location, and the installation procedure is run.

   Proceed to step 6

5B. Multiple-Architecture Installation

If this is an additional platform in a multiple-architecture installation, i.e. the INSTALL script is being run again to add support for an additional architecture, (refer to the Multiple Architecture Support Note in Step 1 of "Starting the installation procedure"), proceed as follows:

The script asks you to verify the information entered, prompting with:

Continue [yes] ?

Hit return to continue. The script should respond with;

The directory hdfeos already exists.

[O]verwrite, [R]e-use or [Q]uit (default) ?

Type 'R' and hit return. The script will build HDF-EOS for the new architecture using the existing copy of the directory structure. Libraries and executables will be added to the architecture-specific subdirectories of the HDF-EOS 'bin' and 'lib' directories, respectively. Do NOT use the Overwrite option - it will clobber the previous architecture-specific installation(s).

6. This completes the interactive portion of the HDF-EOS installation. When the HDF-EOS section is complete, it outputs the message:

   HDFEOS installation ending at: <date/time>

   For information about user setup, as well as instructions for compiling and linking with HDF-EOS, Refer to the file README in the HDF-EOS 'doc' directory.

Toolkit Installation Section

1A. SCF Installation

If the SCF version of the toolkit is being built (the default), the script outputs the messages:

Running the Toolkit Installation Script

Toolkit installation script: INSTALL-Toolkit

Starting at: <date/time>

The SCF version of the toolkit library libPGSTK.a will be built

1B. DAAC Installation

If the DAAC version of the toolkit is being built (-daac option), the script outputs the messages:

Running the Toolkit Installation Script ...

Toolkit installation script: INSTALL-Toolkit

Starting at: <date/time>

The DAAC version of the toolkit library libPGSTK.a will be

2. The toolkit installation script outputs status messages as it goes, ending with:

   INSTALL-Toolkit completed successfully at <date/time>

   If an error occurred during the installation process, the last message will appear as:

   INSTALL-Toolkit completed with errors at <date/time>

   NOTE: If the installation was run with the -log option, the above messages will appear only in the log file, not on the screen.

3. Wait for completion messages. If no errors were encountered during either HDF or toolkit installation, the final script message is:

   SDP Toolkit installation completed at <date/time>

   Otherwise messages of the following form will appear:

   INSTALL: Error: <error message>

   SDP Toolkit installation canceled

4. Review the installation log.

   Every attempt has been made to trap all possible installation errors and report them at the end of the installation process. Nonetheless, it is a good idea to review the installation log to verify that it completed without errors. If errors were noted, the log can help to identify precisely what went wrong. Please note that some warning messages, (NOT fatal errors), may occur in the course of a normal successful installation run.

5.1.2.5 User Account Setup

Once the toolkit has been installed, the accounts of SDP toolkit users must be set up to define environment variables needed to compile and run code with the toolkit (see parts 2 and 3 of the Notes section 5.1.2.8, below).The type of setup depends on the user's login shell.

1A. C shell (csh) users:

Edit the SDP Toolkit user's .cshrc file to include ONLY ONE of the following two lines:

(EITHER:)

source <SDP-home-dir>/bin/$BRAND/pgs-env.csh

(OR:)

source <SDP-home-dir>/bin/$BRAND/pgs-dev-env.csh where <SDP-home-dir> is the full path of the toolkit home directory, and $BRAND is an architecture-specific value for your host. Please refer to part 2 of the Notes section, below, to determine the correct value.

The script pgs-env.csh sets up all the variables discussed in part 3 of the Notes section, below, and it adds the toolkit bin directory to the user path.

The script pgs-dev-env.csh sets up all of the variables set by pgs-env.csh and adds the toolkit bin directory to the user path. In addition, it automatically sets up the compiler flag variables discussed in part 4 of the Notes section below, to work on any of the system environments listed in part 1 of the Notes section, below.

The environment variables will become available during all subsequent login sessions. To activate them for the current session, simply type one of the two lines listed above, at the Unix prompt.

Note regarding path setup with pgs-dev-env.csh:

The script pgs-dev-env.csh also makes available a variable called pgs_path. This can be added to the user's path to ensure that it accesses the directories necessary for the compilers and other utilities used to generate executable programs. It is not added to the user path by default,because in many cases it adds unnecessary complexity to the user path. To add pgs_path to the user path, modify the SDP Toolkit user's .cshrc file to include the following:

```
set my_path = ($path)                    # save path
source <SDP-HOME-DIR>/bin/$BRAND/pgs-dev-env.csh # PGS setup
set path = ($my_path $pgs_path)                  # add pgs_path
```

INSTEAD OF either of the two options listed at the beginning of this step. Note that it is the user's responsibility to set up his or her own path so that it doesn't duplicate the directories set up in pgs_path. Please also note that the pgs_path is added AFTER the user's path. This way, the user's directories will be searched first when running Unix commands.

1B. Korn shell (ksh) users:

Edit the SDP Toolkit user's .profile file to include ONLY ONE of the following two lines:

(EITHER:)

<SDP-home-dir>/bin/$BRAND/pgs-env.ksh

(OR:)

<SDP-home-dir>/bin/$BRAND/pgs-dev-env.ksh

where <SDP-home-dir> is the full path of the toolkit home directory, and $BRAND is an architecture-specific value for your host. Please refer to part 2 of the Notes section, below, to determine the correct value.

The script pgs-env.ksh sets up all the variables discussed in part 3 of the Notes section, below, and it adds the toolkit bin directory to the user path.

The script pgs-dev-env.ksh sets up all of the variables set by pgs-env.ksh and adds the toolkit bin directory to the user path. In addition, it automatically sets up the compiler flag variables discussed in part 4 of the Notes section below, to work on any of the system environments listed in part 1 of the Notes section, below.

The environment variables will become available during all subsequent login sessions. To activate them for the current session, simply type one of the two lines listed above, at the Unix prompt.

Note regarding path setup with pgs-dev-env.ksh:

The script pgs-dev-env.ksh also makes available a variable called pgs_path. This can be added to the user's path to ensure that it accesses the directories necessary for the compilers and other utilities used to generate executable programs. It is not added to the user path by default, because in many cases it adds unnecessary complexity to the user path. To add pgs_path to the user path, modify the SDP Toolkit user's .profile file to include the following:

```
my_path="$PATH"                                # save path
 <SDP-HOME-DIR>/bin/$BRAND/pgs-dev-env.ksh # PGS setup
PATH="$my_path:$pgs_path" ; export PATH  # add pgs_path
```

INSTEAD OF either of the two options listed at the beginning of this step. Note that it is the user's responsibility to set up his or her own path so that it doesn't duplicate the directories set up in pgs_path. Please also note that the pgs_path is added AFTER the user's path. This way, the user's directories will be searched first when running Unix commands.

1C. Bourne shell (sh) users:

Set up the required toolkit environment variables by appending the contents of the file

<SDP-home-dir>/bin/$BRAND/pgs-env.ksh

or the file

<SDP-home-dir>/bin/$BRAND/pgs-dev-env.ksh

to the end of the SDP Toolkit user's .profile, where <SDP-home-dir> is the full path of the toolkit home directory, and $BRAND is an architecture-specific value for your host. Please refer to part 2 of the Notes section, below, to determine the correct value.

The environment variables will become available during all subsequent login sessions. To activate them, log out and then log back in.

5.1.2.6 File Cleanup

Once the toolkit has been built and tested, you can delete certain temporary files and directories to save some disk space. Note that once these files have been removed, you will need to unpack the original distribution file in order to re-do the installation. To remove these files:

```
cd <SDP-home-dir>/bin/$BRAND
/bin/rm -r tmp                      # delete temp files used in bin
cd <SDP-home-dir>/database
/bin/rm de200.dat                   # delete ephemeris ASCII file
```

If you plan to use the Ancillary (AA) data access tools, you must now install the AA tools data files, located in an additional compressed tar file, which must be downloaded separately. The installation instructions are located in Section 5.1.4, Installation of AA Tools.

5.1.2.7 Rebuilding the toolkit library

The toolkit installation procedure now makes it easy to rebuild the toolkit library without having to re-install the entire toolkit. This may be useful in the event that any problems are encountered during the installation process.

SCF Installation

To rebuild the toolkit library at an SCF site do the following:

Set directory.

```
cd <SDP-home-dir>
```

Type:

```
bin/INSTALL-Toolkit <install-options> -lib_only
```

where <install-options> are the installation options set in step 2 of Starting the Installation Procedure, above.

5.1.2.8 NOTES:

1. The SDP Toolkit was built and tested in a multi–platform environment using the following platforms, operating systems, and compilers:

Table 5–1. SDP Toolkit Development Configuration

Platform	OS	Version	C Compiler	FORTRAN
Sun Sparc	Solaris	2.4	Sun C 3.0.1	Sun FORTRAN 3.0.1
Sun Sparc	SunOS	4.1.4	Sun C 3.0.1	Sun FORTRAN 3.0.1
HP 9000/735	HP–UX	A.09.05	HP C 9.75	HP FORTRAN 9.16
SGI Indigo	IRIX	5.3	SGI C 3.19	SGI FORTRAN 4.0.2
DEC Alpha	OSF1	3.2	DEC C 3.0	DEC FORTRAN 3.6
IBM RS–6000	AIX	3.2.5	IBM C 2.1.4.0	IBM FORTRAN 3.2.2.0
CRAY J932	Unicos	8.0.4.2	Cray C 4.0.3.22	Cray FORTRAN 6.0.4.25
SGI Power Challenge	IRIX64	6.1	SGI C 6.1	SGI FORTRAN 6.1

Notes:

a. SGI and SunOS 4 are also running NAG FORTRAN-90 2.1.

b. Sun OS 4 also runs ADA 1.1

c. Cray is running Cray FORTRAN 90 1.0.3.5.

d. Compilers are provided by platform vendors unless specified.

2. Toolkit architecture type names

   To track architecture dependencies, the toolkit defines the environment variable $BRAND. Following is a list of valid values for this variable, which is referred to throughout this document:

$BRAND	Architecture
cray	Cray UNICOS
dec	DEC Alpha
hp	HP 9000
ibm	IBM RS-6000
sgi	SGI / SGI Power Challenge (old-style 32-bit mode toolkit)
sgi32	SGI Power Challenge (new-style 32-bit mode toolkit)
sgi64	SGI Power Challenge (64-bit mode toolkit)
sun4	Sun: SunOS 4.x
sun5	Sun: SunOS 5.x (Solaris 2.x)

3. In order to use the SDP Toolkit libraries and utilities, a number of environment variables MUST be set up to point to SDP directories and files. These variables are automatically set up in User Account Setup section of the installation instructions. They are listed here for reference:

Table 5–2. Required Directory Environment Variables

Name	Value	Description
PGSHOME	<install–path>/TOOLKIT (where <install–path> is the absolute directory path above TOOLKIT)	top level directory
PGSBIN	${PGSHOME}/bin/($BRAND)	executable files
PGSDAT	${PGSHOME}/lib/database/($BRAND)	toolkit database files
PGSINC	${PGSHOME}/include	include (header) files
PGSMSG	${PGSHOME}/message	SMF message files
PGSLIB	${PGSHOME}/lib/($BRAND)	library files
PGSOBJ	${PGSHOME}/lib/obj/($BRAND)	toolkit object files
PGSRUN	${PGSHOME}/runtime	runtime work files
PGSSRC	${PGSHOME}/src	toolkit source files
PGSTST	${PGSHOME}/test	test area
PGS_PC_INFO_FILE	$PGSRUN/PCF.relA	Process Control File

4. Other toolkit environment variables

   In addition, the makefiles which are used to build the libraries require certain machine–specific environment variables. These set compilers, compilation flags and libraries, allowing a single set of makefiles to serve on multiple platforms. The User Account Setup section of the installation instructions explains how to set them up They are listed here for reference:

Table 5–3. Required Compiler and Library Environment

Name	Description
CC	C compiler
CFLAGS	default C flags (optimize, ANSI)
C_CFH	C w/ cfortran.h callable from FORTRAN
CFHFLAGS	CFLAGS + C_CFH
C_F77_CFH	C w/ cfortran.h calling FORTRAN
C_F77_LIB	FORTRAN lib called by C main
F77	FORTRAN compiler
F77FLAGS	common FORTRAN flags
F77_CFH	FORTRAN callable from C w/ cfortran.h
F77_C_CFH	FORTRAN calling C w/ cfortran.h
CFH_F77	same as F77_C_CFH
F77_C_LIB	C lib called by FORTRAN main

5. For a complete list of the tools provided with this release of the SDP Toolkit, please refer to Section 1, Table 1–2

6. The majority of the SDP Toolkit functions are written in C. These C–based tools include the file cfortran.h, using it to generate machine–independent FORTRAN bindings.

5.1.3 Compiling User Code with the Toolkit

In order to compile your programs in conjunction with the toolkit, certain flags MUST be set on the compiler command lines. These flags vary, depending on the platform type and operating system.

The toolkit includes command files that set up environment variables to simplify the task of compiling with toolkit code. The following subset is relevant for this discussion:

CC	the name of the C compiler (usually cc)
CFHFLAGS	required C compilation flags (ANSI C mode, optimized)
F77	the name of the FORTRAN compiler (usually f77)
F77_CFH	required FORTRAN compilation flags
HDFSYS	a flag used to tell the code what platform is being used
PGSINC	the location of the toolkit include files
PGSLIB	the location of the toolkit library libPGSTK.a
HDFINC	HDF include files
HDFLIB	HDF Library files

To automatically set up these variables for your platform do the following:

for csh users, type:

```
source <TOOLKIT-HOME-DIRECTORY>/bin/pgs-dev-env.csh
```

for ksh users, type:

```
. <TOOLKIT-HOME-DIRECTORY>/bin/pgs-dev-env.ksh
```

where <TOOLKIT-HOME-DIRECTORY> is the location where the toolkit is installed (e.g. /usr/local/PGSTK)

You may then view the settings of these variables with the command:

```
$PGSBIN/pgs-flags
```

NOTE: On some platforms, some of these variables are blank. This is normal—the compile lines given below should work anyway.

Once the variables have been set as indicated above, the following command lines may be used as a guide to compiling your programs with the toolkit.

C to object:

```
$CC -c $CFHFLAGS -D$HDFSYS -I$PGSINC myfile.c
```

C to executable:

```
$CC $CFHFLAGS -D$HDFSYS -I$PGSINC -L$PGSLIB \
        myfile.c -lPGSTK (-l...) -o myfile
```

FORTRAN to object:

```
$F77 -c $F77_CFH myfile.f
```

FORTRAN to executable:

```
$F77 -c $F77_CFH myfile.f $PGSLIB/libPGSTK.a (other libraries ...) \
        -o myfile
```

If the toolkit was built with HDF support included, and your code uses tools that require HDF support, you may use the lines listed below:

C to object:

```
$CC -c $CFHFLAGS -D$HDFSYS -I$PGSINC -I$HDFINC myfile.c
```

C to executable:

```
$CC $CFHFLAGS -D$HDFSYS -I$PGSINC -I$HDFINC -L$PGSLIB
-L$HDFLIB \
        myfile.c -lPGSTK -ldf (-l ...) -o myfile
```

FORTRAN to object:

```
$F77 -c $F77_CFH myfile.f
```

FORTRAN to executable:

```
$F77 -c $F77_CFH myfile.f $PGSLIB/libPGSTK.a $HDFLIB/libdf.a \
        (other libraries ...) -o myfile
```

The important thing in this case is that your code gets linked with the HDF library. You do not need -I$HDFINC unless your C code makes direct calls to HDF.

5.1.4 Installation of AA Tools

This section covers installation of the data files needed to use the Ancillary/auxiliary (AA) data access tools. These files include the Digital Chart of the World and other earth sciences data sets. If you do not plan to use these tools or data sets, it is not necessary to install the files.

These files will require approximately 260 Mb of disk space. They may be installed in any location; i.e., they do not have to be stored under the SDP Toolkit home directory.

The tool PGS_AA_dcw MUST have access to the files contained in the four directories named /soamafr, /sasaus, /noamer, /eurnasia in order to work. These files comprise about 80 Mbytes. The other tools (PGS_AA_2/3DRead PGS_AA_2/3Dgeo, PGS_AA_dem) are designed to work with a large range of gridded data sets. Those in the tar file are samples of data from National Geophysical Data Center (NGDC) which need not be maintained by the user; i.e., the user should delete which ever are not pertinent. These files comprise about 180 Mbytes.

The installation script for the AA tools data files is included as part of the main SDP Toolkit distribution. Due to space constraints, the data files themselves are located in a separate compressed tar file, called SDPTK5.1v1.00-AAdata.tar.Z, which must be downloaded separately.

You must first install the SDP Toolkit BEFORE installing the AA tools data files. The AA tools data files installation requires a disk partition with about 400 Mb of free space.

To install the AA tools data files from the tar file:

a. Run the INSTALL-AAdata script

    1. If you have already modified your login files, as in the toolkit installation instructions, simply type:

        INSTALL-AAdata

    from any directory.

    2. If you haven't yet done this, then proceed by typing the following:

        cd <SDP-home-dir>
        bin/INSTALL-AAdata

    where <SDP-home-dir> is the full path of the toolkit home directory.

b. The script contains a default name for the distribution file containing the AA tools data files. That name should be correct for the current release of the toolkit. The script will display the default distribution file name and prompt the user for an override. If the name is correct, press return to continue. If installing from a different distribution file for any reason, please enter the name and press return.

c. By default, the script looks for the tar file in your current directory and also in <SDP-home-dir>. If the file is found in one of the default locations, the script will continue to the next step. Otherwise, please enter the correct location when the script prompts for it.

d. The script then asks where the AAdata directory will be created. The default is <SDP-home-dir>. If you want it installed elsewhere, please enter the pathname when the script prompts for the location. Otherwise, simply hit return to continue.

e. The script asks you to verify the information entered. Type 'y' and hit return to continue. The contents of the distribution file are then extracted into the specified location. Please note that this is a lengthy process that will probably take somewhere between 0.5 and 1.5 hours, depending on your host.

f. The script then asks if the Process Control files, should be patched so that the PRODUCT INPUT FILES directory is set to point to the AA data directory. The default is yes. If you answer no, you must you must edit the Process Control File yourself, in order for the AA tools to work.

g. The script then asks if the distribution file should be removed. The default is no. Once you are satisfied that the files have successfully been installed, you will probably want to get rid of this file, as it takes up a lot of disk space.

   If you wish to get a listing of the files contained in the distribution file, for verification purposes, follow the steps below. Please be aware that this is no small task, as there are literally thousands of data files contained in the distribution file. To see the listing, go to the directory where the distribution file is located and type.

   zcat SDPTK5.1v1.00-AAdata.tar.Z | tar tvf -

   You may wish to pipe the output to the UNIX 'more' command, to allow you to see a screen at a time.

   zcat SDPTK5.1v1.00-AAdata.tar.Z | tar tvf - | more

This completes the installation of the AA tools data files.

5.2 Instructions on Making Changes to Installation Procedures

The installation procedures given in the previous subsection should work seamlessly for a platform in Table 5–1. This subsection gives instructions on making changes to the installation procedure of subsection 5.1, which may be necessary if one uses a different configuration. Here we give a step–by–step procedure for making these modifications.

In the following procedure, <SDP-home-dir> refers to the SDP Toolkit home directory.

a. After unpacking the tar file, but before running bin/INSTALL, (steps a–e in Section 5.1, corresponding to steps 1–5 in <SDP-home-dir>/README), edit the file INSTALL in <SDP-home-dir>/bin.

   The section starting with the comment at line #176 and ending at line 330 must be modified for your platform. This section consists of a switch block that checks the value of the environment variable BRAND and sets the flags for each platform accordingly. Modify ONLY the block associated with your platform.

   The proper block can be determined from the following table:

Table 5–4. Values of OSTYPE

value of $BRAND	platform type
sun5	Sun Sparc (SunOS 5.X)
sun4	Sun Sparc (SunOS 4.X)
hp	HP 9000
dec	DEC Alpha
sgi	SGI Indigo
ibm	IBM RS–6000
cray	Cray

Within each block the following variables are set:

Table 5–5. Environment Variables

Name	Description
CC	C compiler
CFLAGS	default C flags (optimize, ANSI)
C_CFH	C w/ cfortran.h callable from FORTRAN
CFHFLAGS	CFLAGS + C_CFH
C_F77_CFH	C w/ cfortran.h calling FORTRAN
C_F77_LIB	FORTRAN lib called by C main
F77	FORTRAN compiler
F77FLAGS	common FORTRAN flags
F77_CFH	FORTRAN callable from C w/ cfortran.h
F77_C_CFH	FORTRAN calling C w/ cfortran.h
CFH_F77	same as F77_C_CFH
F77_C_LIB	C lib called by FORTRAN main
HDFSYS	system type as defined by HDF

Modify the code to set these variables to the appropriate values for your compilers. Variables CFHFLAGS, CFH_F77, and HDFSYS should never require modifications. The most important ones are:

CC the C compiler

F77 the FORTRAN compiler

CFLAGS MUST set the C compiler for ANSI C code

C_CFH needed to compile C toolkit code that uses cfortran.h for FORTRAN bindings

F77_CFH needed when compiling FORTRAN to object code callable from C using cfortran.h

F77_C_CFH needed when compiling FORTRAN drivers that call C subroutines with FORTRAN bindings written in C using cfortran.h

These flags MUST be properly set in order to build the SDP toolkit.

b. edit the file pgs-dev-env.csh.tmp in <SDP-home-dir>/bin/tmp

The section starting with comment at line #84 and ending at line #185 is identical to the previously mentioned section in the file bin/INSTALL, and must be modified in the same way.

c. continue with the SDP Toolkit installation by running bin/INSTALL (step f in Section 5.1, corresponding to step 6 in <SDP-home-dir>/README).

5.3 Link Instructions

This subsection gives instructions on how to link SDP Toolkit libraries with your code.

The delivery consists of a single SDP Toolkit library called libPGSTK.a.

Here we give generic command lines for linking with this library We use $C_COMPILER and $F77_COMPILER to indicate both the compiler name and any machine–specific compiler flags used by the science software developer. The relevant environment variables must have been previously set up; see the "Installation Procedures" subsection of this section.

To link C code in file "main.c" with the toolkit, on all machines:

```
$C_COMPILER -I$PGSINC -L$PGSLIB main.c -lPGSTK -lm
```

To link FORTRAN 77 code in file "main.f" with the toolkit, on all machines:

```
$F77_COMPILER main.f $PGSLIB/libPGSTK.a
```

NOTES:

Specific examples on how to link particular Toolkit functions on the Toolkit development platforms are given with the separately supplied tool test drivers. See the "Test Drivers" in Section 5.4.

If you are using a different development configuration than one of those given in table 5–1 ("SDP Toolkit Development Configuration") of Section 5.1, see Section 5.2 ("Instructions on Making Changes to Installation Procedures") above.

To ensure compatibility of code at the DAACs, science teams are strongly encouraged to use the same compiler switches used by the SDP Toolkit where possible. These switches enforce ANSI/POSIX standards, necessary for compiling the toolkit with the same functionality on all tested platforms; using the same switches in your code makes it more likely that your code will quickly pass integration and test at the DAAC. The compilers and their respective switches are represented by the environment variables **$CC**, **$CFLAGS**, **$F77**, **$F77FLAGS**, and are defined in the file $PGSHOME/bin/pgs_dev_env.csh. **$CC** and **$F77** contain the names of the C and FORTRAN compilers respectively. **$CFLAGS** and **$F77** flags contain the compiler switches (options) used by the SDP Toolkit with the C and FORTRAN compilers respectively.

5.4 Test Drivers

Also included with this toolkit delivery is a tar file containing test driver programs.

These test programs are provided to aid the user in the development of software using the toolkit. The user may run the same test cases as included in this file to verify that the toolkit is functioning correctly. These programs were written to support the internal test of the toolkit and are not an official part of the Toolkit delivery; users make use of them at their own risk. No support will be provided to the user of these programs. The tar file contains source code for a driver in C and FORTRAN for each tool; readme files explaining how to use each driver; sample output files; and input files and/or shell scripts, where applicable.

The UNIX command

```
zcat SDPTK5.1.1v1.00_TestDrivers.tar.Z | tar xvf -
```

will create a directory called test_drivers beneath the current directory containing all these test files.

5.5 User Feedback Mechanism

The mechanism for handling user feedback, documentation and software discrepancies, and bug reports follows:

a. An account at the ECS Landover facility has been set up for user response:

   pgstlkit@eos.hitc.com

b. Users will e–mail problem reports and comments to the above account. A receipt will be returned to the sender. A workoff plan for the discrepancy will be developed and status report issued once a month. Responses will be prioritized based on the severity of the problem and the available resources. Simple bug fixes will be turned around sooner, while requested functional enhancements to the Toolkit will be placed in a recommended requirements data base (RRDB) and handled more formally.

c. The following format will be used for email response. It can be found in the tar file in the SDP Release A Toolkit 5.1 delivery package.

   Name:

   Date:

   EOS Affiliation (DAAC, Instrument, Earth Science Data and Information System (ESDIS), etc.):

   Phone No.:

   Development Environment:

   Computing Platform:

   Operating System:

   Compiler and Compiler Flags:

   Tool Name:

   Problem Description:

   (Please include exact inputs to and outputs from the toolkit call, including error code returned by the function, plus exact error message returned where applicable.)

   Suggested Resolution (include code fixes or workarounds if applicable):

d. In addition to the email response mechanism, a phone answering machine is also provided. The telephone number is: 301–925–0781. Calls will be returned as soon as possible. We note that the email user response mechanism has been in operation for several years and has been effective in gathering user feedback. Email is our preferred method of responding to users.

e. A list of Frequently Asked Questions (FAQ) for Toolkits is also available.

   The URL for the SDP Toolkit Frequently Asked Questions (FAQ) page is http://newsroom.hitc.com/sdptoolkit/faq.html

   You can also get there from the EDHS Home Page http://edhs1.gsfc.nasa.gov/. Click on the "Toolkit Frequently Asked Questions (FAQ)" link.

This page intentionally left blank.